Embedded Internet and the Internet of Things
WS 12/13

7. Transport Layer

Prof. Dr. Mesut Güneş
Distributed, embedded Systems (DES)
Institute of Computer Science
Freie Universität Berlin
Overview

• Introduction
• Refresher: TCP
• RMST
• PSFQ
• ESRT
• STCP
• Summary
Introduction
Introduction

- Transport protocols are used to ...
  - orderly transmission
  - eliminate or mitigate congestion
  - reduce packet loss
  - provide fairness in bandwidth allocation
  - guarantee end-to-end reliability
Introduction

• Transport protocols are used to …
  • orderly transmission
  • eliminate or mitigate congestion
  • reduce packet loss
  • provide fairness in bandwidth allocation
  • guarantee end-to-end reliability
Distinct features of WSNs

• Topology
  • multi-hop star topology

• Diverse applications
  • event-driven
  • continuous delivery
  • query-driven delivery
  • hybrid delivery

• Traffic characteristic
  • Mainly from sensors to sink -> MP2P

• Resource constraint
  • Limited resources (CPU, Memory, Bandwidth, …)

• Small message size
  • No segmentation of messages
Typical traffic structure

- **Sensor -> Actor**
  - Unicast
  - Challenge: Reliability

- **Sink -> Sensor/Actor**
  - Data
    - Control and signaling
    - Code update -> reprogramming
  - Multicast/Broadcast
  - Challenge: Reliability

- **Sensor -> Sink**
  - Data
    - Sensor measurements/events
  - Concast
  - Challenge: Reliability + Congestions
Design criteria for transport protocols

- Performance metrics
- Congestion control
- Loss recovery
Design criteria for transport protocols

[Diagram showing the network layers: Physical, Data Link, Network, Transport, Application, Host, Router]
Performance metrics

• **Energy efficiency** -> maximize system lifetime

• **Reliability**
  • Packet reliability -> consider every packet
  • Event reliability -> not all packets need reliability, but the event

• **QoS metrics** -> as in traditional networks
  • Bandwidth
  • Delay
  • Packet loss rate

• **Fairness** -> distance of the nodes to sink
Congestion control

• Two main causes for congestion
  • Packet arrival rate exceeding packet service rate
    • More likely at nodes close to the sink
  • Link level performance such as contention, interference and bit error

• Three methods involved
  • Congestion detection
  • Congestion notification
  • Rate adjustment
Congestion control

• Congestion detection
  • queue length
  • packet service time
  • ratio between packet service time and packet inter-arrival time at an intermediate node
  • channel loading

• Congestion notification -> inform the nodes
  • explicit notification
  • implicit notification

• Rate adjustment
  • Nodes adjust their traffic sending rate
  • AIMD (Additive Increase Multiplicative Decrease) schemes
Loss recovery

• Loss detection and notification
  • Usual approach -> sequence numbers
  • End-to-end -> like in TCP
  • Hop-by-hop -> receiver based or sender based

• Retransmission-based loss recovery
  • End-to-end -> like in TCP, expensive
  • Hop-by-hop -> nodes need to cache
    • Energy efficient
Design criteria for transport protocols

Application Layer -> Transport Layer -> Network Layer -> Data Link Layer -> Physical Layer

End-to-End
Refresher: TCP
TCP basics

- TCP applies a sliding window flow control

- Congestion window (cwnd)
  - Represents the maximum amount of outstanding data that have not been ACKed
  - The TCP sender maintains an updated value of the window
  - Its value is set by the congestion control algorithm

- Receiver window (rwnd)
  - \( cwnd \leq rwnd \)
    - The amount of data which are not ACKed, cannot be higher than the receiver window (rwnd, set by receiver)

- When an ACK is received, the window slides forward
TCP Congestion Control

• Executed by the TCP sender
• Avoids the collapse of the network due to an overload by the traffic sources
• TCP congestion control is composed by the following algorithms:
  • Slow Start
  • Congestion Avoidance
  • Fast Retransmit
  • Fast Recovery

• All based on the concept of “congestion event”
Start with one segment

Congestion Avoidance, groping to the maximum capacity

Overload assumed, reduce the data amount

Be more careful in the next attempt

Slow start, fast utilization of the free capacity
Congestion Events

- Loss of a segment = congestion

- A segment loss is detected by means of duplicate ACKs (generally ndup = 3)

- A timeout (equal to a certain value RTO) elapses without receiving any ACK for a segment
Slow Start

• Slow start is triggered
  • At the beginning of a new connection
  • When a timeout elapses

• The value of the cwnd is set equal to 1; then it is increased by 1 each time an ACK is received

• Slow Start is terminated when
  • A congestion event occurs
  • The congestion window cwnd reaches a threshold value ssthresh

• During the Slow Start the congestion window cwnd increases exponentially (it doubles at each round trip time)
Congestion Avoidance

- The congestion window $cwnd$ is increased by $1/cwnd$ each time an ACK is received.

- The procedure is terminated when a congestion event is detected.

- During congestion avoidance, the congestion window increases linearly (it increases by 1 at each round trip time).
Fast Retransmit

- It is executed when ndup duplicate ACKs are received
- The sender sends again the segment for which several ACKs have been received
- The Fast Retransmit is terminated immediately and Fast Recovery is initiated
Fast Recovery

- Follows the Fast Retransmit

- The congestion window \( cwnd \) is halved

- The congestion window \( cwnd \) is increased by 1 each time an ACK is received

- When the missing segment is received, the congestion window \( cwnd \) is set equal to the value it had at the beginning of the Fast Recovery (after it has been halved)
TCP Congestion Control Recap

Prof. Dr. Mesut Günes • 7. Transport Layer
Summary: TCP

• Why not TCP?
  • TCP is unable to distinguish between congestion losses and transmission losses → poor performance
  • TCP is not suitable for wireless multi-hop
  • TCP provides 100% reliability → not required by many applications
  • TCP is connection-oriented → expensive and not necessary
  • TCP depends on network-wide unique addresses of nodes
  • Preprocessing or aggregation of data in intermediate nodes is desirable and often necessary
    • Packets can be combined or changed before they reach destination
  • TCP is not light-weight → too much overhead

• Why not UDP?
  • No reliability at all
Challenges for the Transport Layer in WSNs

• Reliability
• Congestion Control
• Self-configuration
  • Adaptive to dynamic topologies caused by node mobility/failure/ temporary power-down, and random node deployment

• Energy Awareness
• Biased Implementation
  • Mainly run on the sink with minimum functionalities at sensor nodes

• Constrained Routing/Addressing
  • No assumption of the existence of an end-to-end global addressing
WSN transport protocols
WSN transport protocols

Transport protocols for WSNs

Congestion control protocols
- STCP [1]
- CCF [6]
- Siphon [11]
- Fusion [4]
- PCCP [7]
- Trickle [12]
- CODA [5]
- ARC [8]

Reliability guarantee
- Upstream
- STCP [1]
- RMST [13]
- ESRT [2]
- RBC [14]
- DTC [9]
- Downstream
- GARUDA [3]
- PSFQ [10]
WSN transport protocols

• RMST (Reliable Multi-Segment Transport)
  • Reliability

• PSFQ (Pump Slowly Fetch Quickly)
  • Reliability

• ESRT (Event-to-Sink-Reliable Transport)
  • Reliability + Congestion
Effect of retransmissions
Effect of retransmissions

• Effect of the number of retries on the resultant probability that a packet will arrive between end points
• Independent loss probability $p$ of transmissions
• Single-hop transmission with up to $R$ retries
• Probability of success $p_R$

$$p_R = \sum_{i=0}^{R-1} p(1-p)^i$$

$$= 1 - (1-p)^R$$

• Consider $h$-hop end-to-end transmission probability $p_e$ with each hop success probability $p_R$

$$p_e = p_R^h$$
Effect of retransmissions

![Graph showing the effect of retransmissions](graph.png)
Effect of retransmissions

• Where to realize retransmissions?

• Transport Layer -> End-to-end

• MAC Layer -> Hop-by-hop
Effect of retransmissions

Figure 2: Probability of arrival across 6 hops
Effect of retransmissions

Figure 3: Number of transmissions required to send 10 fragments across 10 hops. Hop-by-hop vs. End-to-End repair.
Reliable Multi-Segment Transport (RMST)
Reliable Multi-Segment Transport (RMST)

- End-to-end data-packet transfer reliability
- Each RMST node caches the packets
- When a packet is not received before the so-called WATCHDOG timer expires, a NACK is sent backward
- The first RMST node that has the required packet along the path retransmits the packet
- In-network caching brings significant overhead in terms of power and processing
- Relies on Directed Diffusion
• Placement of reliability for data transport at different layers of the protocol stack

• Reliability in the
  • MAC layer
  • Transport layer
  • Application layer
  • Combinations of above

• Trade-off between implementation on the MAC vs Transport layer
RMST

• Implemented as a filter for Directed Diffusion

• Takes advantage of Directed Diffusion for
  • Routing
  • Path recovery and repair

• Adds to Directed Diffusion
  • Fragmentation/reassembly management
  • Guaranteed delivery
RMST

• RMST as a filter
• Transport layer pushed onto the diffusion stack
• Above the gradient filter
Placement of reliability for data transport

• Receivers responsible for fragment retransmissions
  • Receivers are not necessarily end points
  • Caching or non-caching mode determines classification of node

• Two types of loss detected by a “receiver”
  • A “hole” in a sequence of fragments
  • A truncated sequence

• RMST considers 3 layers
  • MAC
  • Transport
  • Application

Prof. Dr. Mesut Günes • 7. Transport Layer
MAC Layer Choices

• No ARQ
  • All transmissions are broadcast
  • No RTS/CTS or ACK
  • Reliability deferred to upper layers
  • Benefits: no control overhead, no erroneous path selection

• ARQ always
  • All transmissions are unicast
  • RTS/CTS and ACKs used
  • One-to-many communication done via multiple unicasts
  • Benefits: packets traveling on established paths have high probability of delivery

• Selective ARQ
  • Use broadcast for one-to-many and unicast for one-to-one
  • Data and control packets traveling on established paths are unicast
  • Route discovery uses broadcast
Transport Layer Choices

• **End-to-End Selective Request NACK**
  • Loss detection happens only at sinks (endpoints)
  • Repair requests travel on reverse (multi-hop) path from sinks to sources

• **Hop-by-Hop Selective Request NACK**
  • Each node along the path caches data
  • Loss detection happens at each node along the path
  • Repair requests sent to immediate neighbors
  • If data is not found in the caches, NACKs are forwarded to next hop towards source
Application Layer Choices

• End-to-End Positive ACK
  • Sink requests a large data entity
  • Source fragments data
  • Sink keeps sending interests until all fragments have been received
  • Used only as a baseline
RMST

• NACKs triggered by
  • Sequence number gaps
    • Watchdog timer inspects fragment map periodically for holes that have aged for too long
  • Transmission timeouts
    • “Last fragment” problem

• NACKs propagate from sinks to sources
  • Unicast transmission
  • NACK is forwarded only if segment not found in local cache
  • Back-channel required to deliver NACKs to upstream neighbors
RMST: Summary

• ARQ helps with unicast control and data packets
  • In high error-rate environments, routes cannot be established without ARQ

• Route discovery packets should not use ARQ
  • Erroneous path selection can occur

• RMST combines a NACK-based transport layer protocol with S-ARQ to achieve the best results
Pump Slowly, Fetch Quickly (PSFQ)
Pump Slowly, Fetch Quickly (PSFQ)

- Pace the data from a source node at a relatively low speed to allow intermediate nodes to fetch missing data segments from their neighbors
  - Assumption: no congestion, losses due only to poor link quality
- Hop-by-hop recovery
- Goals
  - Recover from losses locally
  - Ensure data delivery with minimum support from transport infrastructure
  - Minimize signaling overhead for detection/recovery operations
  - Operate correctly in poor link quality environments
  - Provide loose delay bounds for data delivery to all intended receivers
PSFQ

• Three basic operations
  • pump
  • fetch
  • report

• Alternate between multi-hop forwarding when low error rates and store-and-forward when error rates are higher
PSFQ

• **Pump**
  • Node broadcasts a packet to its neighbors every $T_{\text{min}}$
  • Neighbors who receive the packet check against their local cache discarding any duplicates
  • If it is just a new message the packet is buffered and the Time-To-Live (TTL) field in the header is decreased by 1
  • If TTL is not zero and there is no gap in the sequence number the packet is scheduled for transmission within a random time $T_{\text{tx}}$

\[ T_{\text{min}} < T_{\text{tx}} < T_{\text{max}} \]

• The random delay allows a downstream node to recover missing segments before the next segment arrives from an upstream node
• It also allows reducing the number of redundant broadcasts of the same packet by neighbors
PSFQ: Pump operation

If not duplicate and in-order and TTL not 0
Cache and schedule for forwarding at time $t$ ($T_{\text{min}} < t < T_{\text{max}}$)
PSFQ: Fetch operation

• Fetch
  • A node goes into fetch mode when a sequence number gap is detected
  • In fetch mode a node aggressively sends out NACK messages to its immediate neighbors to request missing segments
  • It is very likely that consecutive packets are lost -> a “window” is used to specify the range of missing packets
  • A node that receives a NACK message checks the loss window field against its cache. If found the packet is scheduled for transmission at a random time in $(0, T_r)$
  • Neighbors cancel a retransmission when a reply for the same segment is overheard
  • NACK messages are not propagated to avoid message implosion
  • “proactive fetch” mode to take care of situations, where the last segment of a message is lost.
  • Node sends a NACK for the remaining segments when they have not been received after a time period $T_{pro}$
PSFQ: Fetch mode (Recovery from Errors)
PSFQ: Fetch quickly
PSFQ: Report

- Used to provide feedback data of delivery status to source nodes
- To minimize the number of messages, a report message travels back from a destination node to the source nodes
- Intermediate nodes can piggyback their report messages in an aggregated manner
Figure 4. Sensor network in a building. A user node at location 0 injects 50 packets into the network within 0.5 seconds.
PSFQ: Results
PSFQ: Results

Testbed Result

Prof. Dr. Mesut Günes • 7. Transport Layer
PSFQ: Summary

- Lightweight and energy efficient
- Simple mechanism
- Scalable and robust
- Need to be tested for high bandwidth applications
- Cache size limitation
- Does not address congestion control
Event-to-Sink Reliable Transport (ESRT)
Event-to-Sink Reliable Transport (ESRT)

- In a typical sensor network application the sink node is only interested in the collective information of the sensor nodes within the region of an event and not in any individual sensor data.
- What is needed is a measure of the accuracy of the information received at the sink, i.e., event-to-sink reliability.
ESRT

- The basic assumption is that the sink does all the reliability evaluation using parameters that are application dependent.
- One such parameter is the decision time interval $\tau$.
- At the end of the decision interval the sink derives a reliability indicator $r_i$ based on the reports received from the sensor nodes.
- $r_i$ is the number of packets received in the decision interval.
- If $R$ is the number of packets required for reliable event detection then
  \[ r_i > R \]
  is needed for reliable event detection.
- There is no need to identify individual sensor nodes but instead there is the need to have an event ID.
- The reporting rate, $f$, of a sensor node is the number of packets sent out per unit time by that node.
- The ESRT protocol aims to dynamically adjust the reporting rate to achieve the required detection reliability $R$ at the sink.
ESRT

$r$ versus $f$ based on simulation results

$n = \text{number of source nodes}$

- $n=81$
- $n=90$
- $n=101$

for $f > f_{\text{max}}$ the reliability drops because of network congestion

$r$ increases with the source reporting rate $f$
ESRT: Protocol overview

- The algorithm runs on the sink
- Sensor nodes:
  - Listen to sink broadcasts and update their reporting rates accordingly
  - Have a simple congestion detection mechanism and report to the sink
- The sink:
  - Computes a normalized reliability measure $\eta_i = r_i / R$
  - Updates $f$ based on $\eta_i$ and if $f > f_{\text{max}}$ or $f < f_{\text{max}}$ in order to achieve the desired reliability
  - Performs congestion decisions based on feedback reports from the source nodes
- Congestion detection:
  - Uses local buffer level monitoring in sensor nodes
  - When a routing buffer overflows the node informs the sink by setting the congestion notification bit in the header packets traveling downstream
ESRT: Network states

Optimal operating region

Optimal operating point \( P_1 = (1, f^*) \)

Required reliability

(No congestion, High reliability)

(No congestion, Low reliability)

(Congestion, High reliability)

(Congestion, Low reliability)
## ESRT: Frequency Update

<table>
<thead>
<tr>
<th>State</th>
<th>f update</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NC, LR)</td>
<td>( f_{i+1} = \frac{f_i}{\eta_i} )</td>
<td>Multiplicative increase ( f ) to achieve required reliability as soon as possible</td>
</tr>
<tr>
<td>(NC, HR)</td>
<td>( f_{i+1} = \frac{f_i}{2 \left(1 + \frac{1}{\eta_i}\right)} )</td>
<td>Decrease ( f ) conservatively, reduce energy consumption and not lose reliability</td>
</tr>
<tr>
<td>(C, HR)</td>
<td>( f_{i+1} = \frac{f_i}{\eta_i} )</td>
<td>Aggressively decrease ( f ) to relieve congestion as soon as possible</td>
</tr>
<tr>
<td>(C, LR)</td>
<td>( f_{i+1} = f_i^{(\eta_i/k)} )</td>
<td>Exponential decrease. ( k ) is the number of successive decision intervals spent in state (C, LR)</td>
</tr>
<tr>
<td>OOR</td>
<td>( f_{i+1} = f_i )</td>
<td>Unchanged</td>
</tr>
</tbody>
</table>

---

Prof. Dr. Mesut Günes • 7. Transport Layer
ESRT: Summary

- Focus on event-to-sink reliability instead on individual end-to-end reliability
- Congestion control mechanism results in energy savings
- Self-configuration property of the protocol is very valuable for random and dynamic topologies
STCP
Sensor Transmission Control Protocol (STCP)

- STCP provides a
  - generic
  - scalable
  - reliable transport
- Majority of STCP functionalities are implemented at the base station
- Each node might be the source of multiple data flows
  - Different flow type, transmission rate, and required reliability
Before transmitting packets, sensor nodes establish an association with the base station via a Session Initiation Packet.

<table>
<thead>
<tr>
<th>Sequence Number (16)</th>
<th>Flows (8)</th>
<th>Options (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock (32)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow Id #1 (8)</td>
<td>Flow Bit (8)</td>
<td>Trans. Rate (8)</td>
</tr>
<tr>
<td>Flow Id #2 (8)</td>
<td>Flow Bit (8)</td>
<td>Trans. Rate (8)</td>
</tr>
<tr>
<td>Flow Id #N (8)</td>
<td>Flow Bit (8)</td>
<td>Trans. Rate (8)</td>
</tr>
</tbody>
</table>
• Sequence number: zero for the session initiation packet
• Flows: number of flows originating at the node
• Clock: Clock value at the time of transmission
• Flow Id: Identify packets of a flow
• Flow Bit: Specifies whether the flow is continuous or event-driven
• Transmission rate: For continuous flows indicates the rate at which a packet will be transmitted by the source node
• Reliability: Expected reliability required by the flow
• **STCP Data Packet Header**

<table>
<thead>
<tr>
<th>Sequence Number (16)</th>
<th>Flow Id (8)</th>
<th>CN (1)</th>
<th>Options (7)</th>
<th>Clock (32)</th>
</tr>
</thead>
</table>

• **CN**: Congestion notification

• **STCP Acknowledgement Packet**

<table>
<thead>
<tr>
<th>Sequence Number (16)</th>
<th>Flow Id (8)</th>
<th>CN (1)</th>
<th>ACK / NACK (1)</th>
<th>Options (6)</th>
</tr>
</thead>
</table>
Continuous Flows

• Sink knows the transmission rate -> can compute the expected arrival time for the next packet
• Sink sends NACK if packet is not received
• Estimated trip time (ETT) based on received packets

• Expected time calculation
  • Timeout is calculated by the expression \((T + \alpha \times ETT)\), where \(T\) is the time between successive transmissions and \(\alpha\) is a positive integer that varies with ETT
  • The second approach is Jacobson/Karels algorithm which considers the variance of the trip time. ETT is used instead of Round trip time.
Event-driven Flows

• The source node buffers each transmitted packet till an ACK is received.
• When an ACK is received, the corresponding packet is deleted from the buffer.
• The nodes maintain a buffer timer that fires periodically.
• When the timer fires, packets in the buffer are assumed to be lost and are retransmitted.
Congestion Detection and Avoidance

• STCP adopts RED with explicit congestion notification with some modification
• Each STCP data packet has a congestion notification bit in its header.
• Every sensor node maintains two thresholds in its buffer: $t_{\text{lower}}, t_{\text{higher}}$
• When the buffer reaches $t_{\text{lower}}$, the congestion bit is set with a certain probability.
• When the buffer reaches $t_{\text{higher}}$, the node will set the congestion notification bit in every packet it forwards.
• On receiving this packet, the base station informs the source of the congested path by setting the congestion bit in the acknowledgement packet
Results

Fig. 7. Average packet latency
Fig. 8. Energy consumption for a 100 node network
## Summary

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Error Recovery</th>
<th>Reliability Direction</th>
<th>Congestion Ctrl</th>
<th>MAC/ Routing Requirement</th>
<th>Sim</th>
<th>OS</th>
<th>Purpose</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMST</td>
<td>h-h/e-e</td>
<td>event to sink</td>
<td>--</td>
<td>DF</td>
<td></td>
<td>ns2</td>
<td></td>
<td>evaluate tradeoffs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ns2</td>
<td></td>
<td>total bytes</td>
</tr>
<tr>
<td>PSFQ</td>
<td>h-h</td>
<td>sink to sensors</td>
<td>--</td>
<td>Broadcast</td>
<td>ns2</td>
<td>tos</td>
<td></td>
<td>compare SRM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>avg delivery ratio/overhead/ latency</td>
</tr>
<tr>
<td>ESRT</td>
<td>--</td>
<td>collective event to sink</td>
<td>event report frequency</td>
<td>CSMA/CA</td>
<td>ns2</td>
<td>--</td>
<td></td>
<td>parameter tuning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>convergence to OOR</td>
</tr>
<tr>
<td>GARUDA</td>
<td>h-h/e-e</td>
<td>sink to sensors</td>
<td>--</td>
<td>Broadcast</td>
<td>ns2</td>
<td>--</td>
<td></td>
<td>loss recovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ns2</td>
<td></td>
<td>Latency/ energy consumption</td>
</tr>
<tr>
<td>MOAP</td>
<td>h-h</td>
<td>sink to sensors</td>
<td>--</td>
<td>broadcast/ uni-cast</td>
<td></td>
<td>tos</td>
<td></td>
<td>Code Updates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Latency/ energy consumption</td>
</tr>
<tr>
<td>CODA</td>
<td>--</td>
<td>collective event to sink</td>
<td>event report frequency</td>
<td>CSMA</td>
<td>ns2</td>
<td>tos</td>
<td></td>
<td>parameter tuning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>energy tax fidelity penalty</td>
</tr>
</tbody>
</table>

h-h = hop by hop  
e-e = end to end  
DF = Directed Diffusion  
tos = Tiny OS
Summary

• Wireless TCP variants are NOT suitable for WSN
  • Resource constraints
• Different notion of reliability
  • End-to-end
  • Hop-by-hop
• Huge buffering requirements
• ACKing is energy draining
Literature
Literature


